Dynamic Predictors for Content Selection in Content Distribution Networks

FABRICE EDGAR NDJAMOY NZOUDJA

Master's Degree Project
Stockholm, Sweden 7 March 2014

Examiners: Carlo Fischione at KTH,
Guido Albertengo at Polito
XR-EE-RT 2014:004
Abstract

Caching in Content Delivery Network is one of the leading methods for saving and providing Quality of Services to users in terms of low latency experienced when requesting multimedia resources. Caching allows a parsimonious use of bandwidth for service providers to have a scalable system and avoid network congestions. Most of the research has focused to save contents in CDN in order to meet the restriction of memory and bandwidth consumption relying on optimal content placement problem and cache policy. The most common policy used to cache content is based on the content’s popularity, i.e., the request frequency. The availability of predictions in the requests of content would allow to optimally cache content. However, how to analyze past content requests to have consistent prediction of future data requests is an open and challenging problem. In this master thesis, this has been addressed by considering data mining, which is a multidisciplinary technique involving theoretical and practical data analysis. Dynamic predictors are designed and proposed to retrieve inherent content information for improving the prediction of the content item selection. Numerical results show that the proposed method achieves good results in term of hit ratio, i.e., low prediction error, which might be used by CDN designer and might be a potential input for the optimal content placement problem.
Acknowledgement

I thank God for all He has provided and still provides me in this beautiful World. I also extend a warm gratitude to my dear family for material and immaterial support it brings me. In addition, I thank the Polytechnic School of Turin, its faculty and its vibrant double degree Erasmus program which has strengthened my integration and enriched my engineering education. I thank KTH and its faculty for the reception and training of quality I received there. I thank Manja Schubert and Magnus Lindqvist our International Coordinators for the quality of services, and the availability they always have granted me. I address also a particular consideration and thankfulness to my supervisors and examiners Carlo Fischione at KTH and Guido Albertengo at the Polytechnic university of Turin who have followed me meticulously and methodically. I address a warm acknowledgement to all my colleagues for all the moments we have shared and enjoyed together.
# Contents

1 **Introduction** 5  
1.1 State-Of-The-Art 5  
1.2 contribution and plan 6  

2 **Caching in Content Delivery Networks** 8  
2.1 Little Evolutionary History 10  
2.2 Content Delivery Networks 11  
  2.2.1 Relationships Between CDN Components: Caching 13  
  2.2.2 Relationships 13  

3 **Data Mining** 15  
3.1 Some applications: Web mining 18  
3.2 Concepts, Instances, and Attributes 19  
3.3 Data Preprocessing 19  
3.4 Classification and prediction 20  
3.5 Algorithms 20  
  3.5.1 Context Specification and Algorithms 21  
  3.5.2 Cross-validation 24  
  3.5.3 Predictor Performance Evaluation 25  

4 **Mathematical Modelling** 26  
4.1 Problem Formulation 26  
4.2 Predictor Modelling and Metric Definition 28  
  4.2.1 Model 1 28  
  4.2.2 Metric Specification 32  
  4.2.3 Linear Model 2 32  
  4.2.4 Metric Specification 33  
4.3 Regression Analysis and Kalman Analogy 33  
  4.3.1 Regression Analysis 33  
  4.3.2 Kalman Filtering Analogy 36
5 Simulation and Results

5.1 Experiments Setting .................................. 38
5.2 Performance Analysis ................................. 40
  5.2.1 Predictor Model 1 ................................. 40
  5.2.2 Predictor Model 2 ................................. 49
  5.2.3 Regression and Kalman Filter .................. 49

6 Conclusion and Future Work

6.1 Goal and Insight .................................... 54
6.2 Future Work ........................................ 55
6.3 Privacy Issue ........................................ 55
Chapter 1

Introduction

Content Delivery Networks have been for the past decade the most efficient networking technology used to overcome the inherent limitations of internet in term of user experienced Quality of Service when requesting Web content. This limitation is even more challenging with the proliferation of smartphones, tablets, which has increased considerably the network traffic. The most common technique used for facing the scalability and improve the performance consists in replication and caching of content in server strategically spread over the world; CDN have been designed for that purpose. Moreover, CDN provide services that improve the network performances by maximizing the bandwidth, improving accessibility and maintaining correctness through content replication [1]; they offers a reliable service by distributing content to cache on edge servers (also called surrogate) located close to the end-user. However, caching is one of the CDN services that designers are still willing to improve in order to boost not only the QoS, but optimize the bandwidth consumption in order to have a system more scalable. Hence, caching is till a problem to tackle.

1.1 State-Of-The-Art

Most of the recent researches have been oriented to how to manage (cache) content items once the replication process has been triggered. Over the last few years, researches have been conducted for understanding caching. Probabilistic in-network caching for formation-centric network has been considered[2]; the authors of this paper approach the caching concept by approximating the cache capability of paths, and caches contents probabilistically in order to reduce cache redundancy, and traffic redundancy. However this study have been conducted on a limited (size) network topology (leaves of a tree) and
does not take into account some network concept for example routing table. In this non-realistic scenario they have experienced good results which perform better than previous proposed schemes such as Leave Copy Down (LCD), and Cache Everything Everywhere (CEE); in term of efficient resource utilization and and fairness in multiplexing the contents along the path. Furthermore, there is also Approximated Model for General Cache Network [3]. The authors of this paper approximate the behaviour of multiple cache network by improving existing Single Cache Algorithms. a-Net the developed algorithm operates in a context close to a realistic scenario by taking into account the network connectivity, the caches size, request stream distribution, and source clustering. They have used LRU policy for cache management and some metric like Miss Probability Ratio (MPR) for the performance analysis of their algorithm. However, this algorithm works only for networks with static routing table and have several issues such as the increasing of the MPR when the Independent Reference Model (non-IRM) is violated. All these papers did not analyse traffic and retrieve some content request stream pattern for improving the caching policy. In this master thesis, following the suggestion of my examiner and supervisor, I am concerned with the prediction of content distribution. These Dynamic predictors perform the selection of content based on inherent information of the multimedia traffic analysis. Hence, this process has to rely in a formal and practical theory.

1.2 contribution and plan

The traffic Analysis may make appeal to data mining framework; which is a framework that provides formal theory and practical tools for pattern retrieval through mining of the data. Hence, the crux of my contribution in this exciting research field of network services consists of integrating the concept of learning in of CDN’s traffic analysis for boosting the way content are cached relying on a formal and practical theory. The motivation of this work relies on caching has been always done by taking into consideration mainly the popularity of content when it comes to cache an item. Hence, this master thesis proposes a new scheme by designing a learner (Dynamic predictors) able to provide a new approach for selecting content to cache. Moreover my work is not user-oriented, i.e., the application of learning in this context takes as input the data log of the users of a CDN and fuses it in a unique data log keeping the continuity over time of requests (end-users are not synchronised). Hence, for concretising and formally developing, the rest of the master thesis report has been structured as follows: Chapter 2 provides an explanation of the importance of caching in CDN. Chapter 3
develops the concept of learning and is followed by algorithms. Chapter 3 formulates the problem of traffic modelling. Chapter 4 develops an attempt to design a mathematical model of the content item request stream. Chapter 5 provides the simulation set up and obtained results. Finally, Chapter 6 concludes the master thesis.
Chapter 2

Caching in Content Delivery Networks

CDN have been for a while a solution for an effective content delivery to end-user. It consists of optimized arrangement of networking and web devices for achieving the above said goal [4]. Collaboration among CDN components can occur over nodes in both homogeneous and heterogeneous environments [1]. They can be centralized, hierarchical or decentralized systems. Moreover, the typical functionalities of a CDN include:

- Request direction and content delivery services to the closest suitable surrogate server using mechanism to bypass congestion, hence overcoming flash crowds or SlashDot effects.
- Content outsourcing and distribution services to duplicate and/or cache content to distributed surrogates servers on behalf of the origin server.
- Content negotiation services to meet specific needs of each individual user (or group of users).
- Management services to manage the network components, to handle accounting, and to monitor and report usage.

The crux of CDN is a continuous improvement of performance through caching or replicating content over some mirrored web server (surrogates servers) strategically placed at various locations in order to deal with the sudden spike in web content requests [1], which is often called flash crowd [5]. All is done in order to redirect the user requests to a closest surrogate (edge servers), which allows to address efficiently the issue of latency faced when providing content to end-users. In the CDN world, a content refers to any multimedia resources, and it consists in two parts: the encoded media
and metadata [6]. The encoded part involves static and dynamic multimedia item (audio, video, text, images and web pages). Metadata is the content description that allows identification, discovery, and management of multimedia and also eases the interpretation of multimedia data [6]. CDN provides an overlay network services relying on application layer protocols such as HTTP or RTSP for transport [7].

![Diagram of CDN](image)

Figure 2.1: Overview of CDN where users represented by mobile devices are connected to base stations and make requests for content items located in origin servers.

The main components of a CDN architecture are: content provider, CDN provider and end-users. A content provider is one who delegates the URI name space of the web objects to be distributed. The origin server of the content provider holds those objects [1]. The CDN provider is a proprietary organization or company that provides infrastructure facilities to content providers in order to deliver multimedia item in time and reliable manner. End-users are the consumers of services of CDN provider. Clients or end-users get access to those services through the content provider’s website, when they make requests of URL. Figure 2.1 gives an overview of CDN in mobile Network.
2.1 Little Evolutionary History

It quite often happens that a given server is overwhelmed by requests which conducts it in a temporarily unavailable state. Clients may experience therefore, a considerable latency when requesting contents; that leaves them in frustration. For instance companies earn significant financial incentives from web-based e-business. Hence, they want their clients to be satisfied when accessing their Web sites. Almost all public services can be done online and most of the software application for multimedia content delivery are web-oriented. This increases the probability to experience a bottleneck. Hence, the past few years have seen the evolution of technologies which aim to improve content delivery and services provisioning over web. Content network was the buzzword used for terming that technology.

Consequently for facing the above mentioned challenges, several content network designers have used different mechanisms to improving the QoS [1]. Some of them have focused their design on modifying the traditional web architecture by improving the Web server hardware adding a high speed processor, more memory and disk space [1]. This is not an efficient solution [8]. Proxy servers deployment is another solution that have been adopted by ISP. Caching proxies consist of placing proxy servers close to users in order to improve the performance of the network in term of bandwidth consumption and latency experienced by end-user when requesting contents. Hence, the user’s entire session is firstly directed to proxy servers and in case the content is not present in proxy servers the request is redirected to origin servers. therefore, the caches contain most popular contents browsed by users. An additional improvement have been achieved by deploying hierarchically caches at different levels which may be local, regional, national or international.

Server farm is another solution which is used to addressing the issue of scalability. A server farm is group of servers in charge of providing content associated to the same web site. It also makes use of a layer 4-7 switch, Web switch or content switch that examine content request and dispatches them among the group of severs [1]. A server farm can also be combined with hierarchical surrogate servers in order to make a more scalable and flexible system [8]. However, this combination has some limitations. Because firstly servers are deployed close the origin servers, which does not solve the congestion problem. thanks to caching proxy servers this burden may be alleviated. Nevertheless, the proxy caching servers contain popular requested content items. Furthermore, in proxy servers are saved only contents which have been requested by end-users. This may force the content providers with popular item sources to invest in large server farm, load balancing, and high bandwidth connections to keep up with demand [1]. For overcoming those
CDN has revolutionized the way contents are stored and delivered. Content providers started putting their web sites on a CDN. Thanks to that new networked technology reliability, scalability, and less expenditure in term of cost of infrastructure have been achieved. One of the giant in that realm Akamai Technologies evolved out of a MIT research effort aimed at tackling the flash crowd issue [9, 10]. Since then, CDN became a huge market for generating revenues. First generation CDNs mostly focused on static or Dynamic web documents. The second generation has shifted to Video-on-Demand (VoD), audio and video streaming. However, it is always in its embryonic phase.

The recent evolution in wireless communication, the explosion of wireless communication devices, and high demand of video content or other multimedia content have made CDN a efficient solution which might be used in mobile network environment as well.

### 2.2 Content Delivery Networks

Figure 2.1 illustrates an architecture of CDN where infrastructure (web servers) are deployed in order to be close to end-users. Content providers can make agreements with CDN proprietaries for storing their contents. This can be done under demand of end-users in that case CDN providers execute pull content operation. On the other hand, content may be replicated on the CDN by its proprietary under demand of the content providers which correspond to a push operation. The latter is done based on a user traffic analysis, where some criteria are considered for pulling or pushing content. Some of them could be either the content popularity feature or the life time of a content (long period of request of content items). A user is served with the content from the nearby replicated web server (CDN server) [1].

CDN providers ensure an efficient (low latency and reliable) delivery of a variety of contents. These are for instance static content (HTML pages, images, document), streaming media (audio, real time video). Content providers are Web service providers, media companies and news broadcasters. Moreover, CDN owners charge content providers according to the type of delivered traffic. The average cost of charging CDN services is high [1] [7]. This is due to the below physical and infrastructural network limitations:

- bandwidth cost
- variation of traffic distribution

limitations another type of content network have been deployed: Content Delivery Networks.
- size of content replicated over surrogate servers
- number of surrogate servers
- reliability and stability of the whole system and security issues of outsourcing content delivery.

Figure 2.2: CDN Composition: CDN organization defines which policy is used for manage requests, servers define the different type of servers in CDN, relationships provide an overview of relation between CDN’s components, inter-cache interaction gives the different type of communication between CDN’s components, and content/service describes the different multimedia items and Services
2.2.1 Relationships Between CDN Components: Caching

CDN incorporates dynamic information about network conditions and load on the cache servers, to redirect request and balance loads among surrogates [1]. It has a flexible structure; it is organized in surrogates which exchange information within them in order to redirect users request for an effective content-delivery and interaction protocols for communication among the CDN components. Figure 2.2 depicts a typical CDN composition.

Most of the commercial CDN providers such as Akamai, AppStream, and Limelight follow the overlay approach of CDN organization [1]; which consists of replicating multimedia items in several servers spread strategically over the globe. The advantage of this type of organization relies on CDN providers do not deal with network electronic devices such as routers switches, hence the management is more simplified. There is also a network approach which relies on networking sophisticated devices (routers and switches with incremental functionalities) for redirecting traffic and dispatching adequately requests among surrogate servers. The servers are of two types: origin and replica servers. The former contains definitive version of a resource. The latter contain copies of resources present in the origin for satisfaction of user requests. In addition, they may serve as web server, media server or as cache server. The media servers deliver multimedia content under request of clients. The web servers contain links to all provided contents. The cache servers contain copies of content, and their role is to reduce as much as possible the client requests directed to origin servers. We are more interested of this type of server. This master thesis focuses on user traffic analysis for improving caching.

2.2.2 Relationships

We are interested on relationship between replica and caching servers of a CDN. Figure 2.3 depicts a representation of functional connection between servers and clients. Replication and caching are the core aims of relation among these constituent components. The replication is more about pushing content from origin servers to replica servers (create and maintain duplicate copies of some contents strategically in different location). Caching is about save a copy of some contents (the most popular ones) in surrogate servers close to clients to reduce significantly the latency and the bandwidth consumption.

User requests may flow from clients to surrogates servers in case the requested content is present in the surrogates, otherwise the request is redirected to origin servers. The communications between clients and surrogates
are implicit, i.e., the clients believe that they are communicating with the origin servers, while they communicate with the surrogates.

Figure 2.3: Client-Surrogate-Origin servers relation
Chapter 3
Data Mining

Our daily life is overwhelmed by data. Economists, statisticians, forecasters, and communication engineers have long worked with the idea that patterns in data can be sought automatically, identified, validated, and used for prediction [11]. All those operations can be grouped under learning concept. Hence, we may want to retrieve from recorded data some patterns that can be used for improving the efficiency of a technology (caching). On the other hand, we can say that Things learn when they change their behaviour in a way that makes them perform better in the future [11]. Furthermore, learning and training are sometimes confusing terminologies and arise philosophical questions. However, they can be combined and put in a same context. The core of this master thesis consists of assessing the performance of predictors which have been design for prediction relying on inherent information of a data log. The inherent information retrieval from the data log have been done thanks to a formal theory of learning process with practical application: data mining.

Data mining is a topic that involves learning in a practical non theoretical sense; it can be used as technique for finding and describing structural patterns in data, which are a tool helping to explain that data and make prediction [11]. Machine learning overlaps with data mining. Machine learning is a burgeoning technology for mining knowledge from data. Data mining is not only knowledge acquisition, but prediction from data, as contrast to learning, which focuses more on structural description of what has been learned.

In the context of data mining, the input data may be a set of examples, e.g, the number of time an item has been requested over time. The output is the prediction, i.e., based on other parameters decide whether or not an item can be valid candidate for pre-caching for the next window time. Furthermore, data mining is also about performance. This can be done through a graphical representation of the predictor performances. Data mining involves
man y disciplines as depicted in the Figure below3.1.

However data mining, can be subject of issue of privacy. Because, sometimes the process of learning can not be extended because of restriction on data. The restriction is caused by sensitivity of data to mine. Data mining may be analogically associated to other terms such as knowledge mining from data, knowledge extraction, and data dredging. The knowledge discovery from data is a continuous process which involves following steps:

1. Data cleaning to remove noise and inconsistent data;
2. Data integration where multiple data sources may be combined
3. Data selection where data relevant to the analysis task are retrieved from the database
4. Data transformation where data are transformed and consolidated into forms appropriate for mining by performing summary or aggregation operations
5. Data mining (an essential process where intelligent methods are applied to extract data patterns)
6. Pattern evaluation to identify the truly interesting patterns representing knowledge based on interestingness
7. Knowledge presentation where visualization and knowledge representation techniques are used to present mined knowledge to users

Figure 3.2 above illustrates the contextualisation of data mining for prediction in Content Delivery Networks. Database may be associated to the users data log repository and data warehouse to the compressed version of the data log (regroup user in per day activity files).

Without loss of generality, data mining can be applied to any type of data. The type of pattern which can be mined from data depend on used functionality. Data mining functionalities include characterization and discrimination; the mining of frequent patterns, associations, and correlations; classification and regression; clustering analysis; and outlier analysis. Data mining functionalities are used to specify the kinds of patterns to be found in data mining tasks. Mining can be either descriptive or predictive. Descriptive mining tasks characterize properties of the data in a target data set. Predictive mining tasks perform induction on the current data in order to make predictions [12]. We are more concerned about classification and regression for predictive mining. Outlier analysis can not be integrated in our topic because, a specific model of user request content is not defined; it can just be approximated by some function (linear in our case), due to the complexity of description of user traffic pattern.
3.1 Some applications: Web mining

The Web mining consists of tracking the users activities on the web not only for improving their navigation but for influencing their behaviour. for instance PageRank metric defines by Google’s founder attempt to measure the standing of page. The more a web site has link more prestige it has; if the pages that link in have high prestige themselves. This allows to easy the research on the web. Moreover, by mining the users queries, search en-
gines can provide correlated advertisement and gain money from advertiser. Companies invest a lot of money in web mining.

3.2 Concepts, Instances, and Attributes

The data under analysis have specific name related to the phase of the learning process and the type of data. We are interested in what is going on in between the input and output. The input takes the form of concepts, attributes, and instances. The thing to be learned is called concept description. It is the most intelligible part, because it can be understood, discussed, disputed, and operational, i.e., it can be applied to the current example. The information given to the learner takes the form of a set of instances. Each instance is characterized by the value of the attributes that measure different aspects of the instance. Data mining deal with numerical, nominal, and categorical attributes. We are more interested on numerical attributes.

There are four different styles of learning applied in data mining. In classification learning, the learning scheme is presented with a set of classified examples from which it is expected to learn a way of classifying unseen examples. In association learning, any association among features is sought, not just ones that predict a particular class value. In clustering, groups of examples that belong together are sought. In numeric prediction, the outcome to be predicted is not a discrete class but a numeric quantity. Independently of the style of learning the thing to be learned is called concept and the output provided by the learning scheme is called concept description.

3.3 Data Preprocessing

Data have to be preprocessed before performing learning. This process aims to ease the retrieving of substantial information from data. Furthermore, it allows to give quality to data. The quality includes accuracy, completeness, consistency, and timeless.

The accuracy is not our concern because the data we are going to deal with do not suffer from any error due to instrumental or computer gathering errors. In addition, the data we are going to deal with do not suffer from any incompleteness, because all attributes have been well reported. It is important to highlight that the quality of data depends on the intended use of the data. In our case, we use data mining for designing predictors. Regarding the consistency aspect, the data log under analysis contains several users reported request during 3 months. The files reporting users activities
have the same format so inconsistency is not our concern as well. Finally, the timeliness aspect is the major problem we faced. Since, users activities are not synchronized we have to put in the same file all the users activities per month. Then, sort the time attribute in order to have continuity; which process will easy the processing of data. Moreover, we are interested on a specific type of content (images and video). Hence, the compression process we use is not complex. We have just removed thanks to a parsing non desired content (e.g, text content). These operations are referred as data cleaning, and data reduction. Moreover, we have transformed the user track activity of the data log by per day request content by removing the user dependency and associating to each item its request frequency per day as well.

3.4 Classification and prediction

Classification is a process that aims to define a model which describes and differentiates concepts. The model are derived based on the analysis of a set of training data (i.e., data objects for which the class labels are known). The model is used to predict the class label of objects for which the class label is unknown test data [12]. Hence classification is a process of predicting categorical labels (discrete). For instance in our case, we would like to design predictors able to provide a list of potential items that may be requested in a very close future based not only on their request frequency history, but on a content request stream model. Hence, our predictors rely on prediction and classification. We are going to deal with two classes: elected and rejected. Elected means that the item is candidate for next window of analysis (pre-cached), otherwise it is not.

3.5 Algorithms

Learning Algorithms are a way of retrieving information from data relying on a model which is tuned during a phase of learning called training. Learning provides techniques to dynamically tune a model relying on a part of data (training data) in order to improve the desired performances of learner tested on another part of the data called test data. Hence, a learning algorithm should consist of two phases which are training, and testing. The model to tune depends on the type of data under analysis and has to match the context of the topic in order to avoid non-significant parameter description during the mathematical modelling. Furthermore, we are going to use supervised learning algorithm. Here supervised means that we have a model of user
request stream and it is tested on the data set. It means that we have a priori some model which is design in Chapter. The learning process may be depicted as shown in Figure 3.3.

![Figure 3.3: Overview of learning process](image)

The illustration it self-explanatory and shows the continuous process of learning through the cycle training-prediction-model. The performance part is implicitly taken into consideration in that cycle. We are going to use the training set for tuning the modelled item request stream.

### 3.5.1 Context Specification and Algorithms

In this thesis, we analyse data from a data log. These data are the track of user activities for 3 months. For instance let us consider $\Sigma$ the set of all possible content items. $\Sigma = \{i_1, i_2, ..., i_n, ...\}$, in every time window we will deal with a subset $\Omega$ of $\Sigma$. $\Omega = \{i_1, i_2, ..., i_y\}$, which is the set of items requested by the users. We define the vector $R$, $R = \{r_1, r_2, ..., r_y\}$, which contains the number of times that each multimedia item has been requested during the current window and the past windows summarized in $r_x$. 
for \( x = 1, \ldots, y \). And \( W \) the set of Windows, \( W = \{w_1, w_2, \ldots\} \). The training phase is performed for instance in \( w_1 \) and the items in \( w_2, w_3, \ldots \) are of two types: the newly requested, and the pre-cached resulting from prediction based on the prediction applied during the training period. Hence, the pre-cached items belong to \( \hat{I} \), because they have been obtained after prediction. The vector \( \hat{I} \) contains all content items that might be requested during the next window time. We define the vector \( \hat{R} \) which contains predicted request frequency of each item in the current items for the next time window. The size of the window of analysis is a fundamental parameter to define (see Section 3.5.2). The algorithms described further are based on three type of predictors. Two of them use the request frequency history and an other parameter resulting from the Minimum Square Error (MSE). All of them are based on linear model and the reason of this choice is explained in Chapter 6. A general skeleton of learning algorithm in the research context is shown in the list below:

Input:

a. Set of item requested for some period of time.

b. Request frequency of relative to each item.

c. Window time of analysis.

Output:

a. Most requested content items.

b. Hit error vector.

c. Predicted content request Frequency.
Algorithm 1 Dynamic Predictor

1: Define $T_{\text{max}}$
2: $\Sigma = \{i_1, i_2, ..., i_n, ...\}$, and $\Omega = \emptyset$
3: $I = \emptyset$
4: $\hat{R} = \emptyset$
5: Get current time
6: while current time $\neq T_{\text{max}}$ do
7:   for $i_x$ request under $W(t) = k * T_{\text{slot}}$ do
8:      Get $\Omega \leftarrow i_x$
9:      to each items in $\Omega$ is associated $r$
10:   end for
11:   Get the $R = \{r_1, r_2, ..., r_y\}$
12:   for $r_x \in R$ do
13:      call model
14:   end for
15:   Get $\hat{R}$
16:   Get $B^*$ optimized (relying on the model).
17:   for $b_{i_x} \in B^*$ do
18:      if $b_{i_x} \geq \Delta$ then
19:         $I \leftarrow i_x$
20:      end if
21:   end for
22:   Check if items predicted belong to window $W_{t+1}$ compute hit rate (predictor type 1 and 2)
23:   Check if the predicted items request frequency in $\hat{R}$ are matched. Compute error.
24:   if predictor type 1 or 2 is used and hit ratio very low then
25:      Change predictor
26:   end if
27:   Add current content set of content item in training set.
28: end while
Algorithm Description

$T_{\text{max}}$ depends on the period of time data have been saved in the data log. In our case, it corresponds to three months of data to analyse. The vector container of elected items and the vector container of predicted request frequency are defined (in line 3-4 Algorithm 1). We define the starting point time of the algorithm. I have chosen the first day of December 1994 as starting point. The aggregation of data relative to a window time ($T_{\text{slot}}$) is done every period of time multiple of $T_{\text{slot}}$ (in line 7-11 Algorithm 1). After aggregation of data, i.e., acquisition of requested items and their relative request frequency (in line 7-11 Algorithm 1) the data are ready for being given as input to predictors (in line 12-14 Algorithm 1). The computation of predicted values based on the modelled predictors is done for each item and some decision are taken for selecting item according to defined criterion (in line 15-21 Algorithm 1). Finally, the performances of predictors are assessed relying on their types (either hit counter or request frequency matching). That is done in the next time window (in line 22-23 Algorithm 1). The process is kept going by following the cross-validation defined in section 3.5.2. until all the data log is mined.

The above algorithm is the summary of three algorithms, which use three different predictors. The mathematical modelling of these predictors is done in Section 4.2.1. Moreover, the window of analysis is defined following the theory of cross-validation. And the parameters relative to each predictor are computed dynamically in order to not lose the matching between data change over time and predicted values. Hence, the sets $\hat{R}$ and $\hat{I}$ are updated during every training and test period.

3.5.2 Cross-validation

Cross-validation is a formal way of dividing data for mining purposes. The adaptation of cross-validation in the context of prediction for Content Delivery Network is possible as well, since we have limited amount of data (three months). Hold out is one of the formal way of organizing data. This method consists of splitting data into three parts in which the two third are used for training and the one third for testing; repeating the process several time with different selection of training and test data. However, it has been shown that the holdout does not give consistent values obtained during training and test [11]; predictors suffer from biasing.

In data mining, the concept of stratification is used for ensuring that a balanced proportion of classes under analysis is present both in the training data and test data; the stratification mitigates the biasing issues. In this
master thesis, we deal with images and video. Hence, this requirement is fulfilled. Since, in each window time we have only one class of content (images) and other classes have been removed during the pre-processing phase.

Furthermore, the k-fold cross-validation is used in order to mitigate the effect of non equal representation of class in the training data and test data. The most used is the 10-fold-cross-validation with consist of dividing the data in 10 parts and choose randomly one of them for test data and the remaining nine-tenths for the training. Then, the error estimate is computed. repeat the procedure 10 times and compute the error as average of the 10 error obtained. NB in order to get consistent result, both training data and test have to be approximately stratified. Even if it has been proved that this method shows good result [12], it suffers from computational cost when applied on huge amount of data. 10-fold-cross validation have been proved to be the most efficient due to its low bias and variance, but other methods may be used such as the leave-one-out cross-validation and the bootstrap. We are not considering this method in this master thesis.

My proposal is to follow the cross-validation concept but instead of dividing the data in 10 parts we are going to use the first 7 days of the data log for training; then, perform the test on the eighth day. Then the data of the eighth day will be added to the training set and computation based predictors will be performed and tested on the ninth day. The process is kept going until we have covered the whole data log. Furthermore, in order to give a chance to every content item to be elected, the algorithm keep the content item list growing, i.e., even if an item is not elected after prediction, it is saved in the content item list. This intuitive idea (see Section 4.1 for elucidation) allows also to keep in touch with the reality, because some item may be requested even if they did not have an interesting past (less or not at all requested in the past). Hence, relying on the cross-validation described above the $T_{\text{slot}}$ defined in the algorithms above is the time corresponding to one day.

3.5.3 Predictor Performance Evaluation

A common way to evaluate the performance of a predictor consists of computing the Means Square Error MSE between the predicted value and the real value. Several other metrics can be used, but we are going to focus on MSE error, and plot the evolution over time of the error in chapter 5. We are going to use three predictors. Two of them combine prediction and classification and just one of them makes prediction of content request frequency. The proposed modelling part relies on linear model. The reason of this particular model flavour is explained further.
Chapter 4

Mathematical Modelling

In this chapter, we develop a model of the predictor we intend to propose in this master thesis. Before starting the modelling, we give a clear formulation of the problem we intend to study in this chapter.

4.1 Problem Formulation

The prediction of content popularity is very complex because it involves human being browsing behaviour, and by consequence it does not follow a known physical law. The users content request stream has almost chaotic evolution over time. In addition, the law of popularity which suggests that if an item has high request frequency one day it has high probability to be requested the next day is not always valid. This is loosely speaking the spirit behind the zipf-mandelbrot function, which is one of the most used density function when concerning popularity of the request of content. Figures 4.1 below shows the item request of three months of users request traffic (only images and video).
Figure 4.1: Three months Item request.

The plot of Figure 4.1 is relative to the max item request frequency per day (figure on the right). The figure on the left is a logarithmic plot of content item frequency. We can infer from the figures the complexity of the track. A statistical analysis of the life time of items reveals that 98% of item out of 3964 contents retrieved from the data log have less than 7 days of life time. That is an indicator of how the data under analysis makes difficult performing a prediction. On the other hand, if the track was relative to an item we may have inferred linear behaviour of the item per days. This simple idea gives a tip on what may be the model associated to the live time of an item. Figure 4.2 illustrates at least 21 days living content, i.e., content items which have more than 21 days of live time. This plot shows the lack of sufficient information to track data.
Figure 4.2: Track of items with live time of at least 21 days.

The goal is to design a predictor that follows with good approximation not only item track, but the election of item for next day. Ideally, the predictor should have low variance in prediction error and should not be biased.

4.2 Predictor Modelling and Metric Definition

In the previous section, we have shown the difficulty in the prediction of the item requests. Without loss of generality, the aim of the prediction is to improve the performance of CDN which can be achieved by pre-caching contents once have been selected after prediction. In the following, we present a modelling of the state to be predicted, and later some predictor proposals.

4.2.1 Model 1

Let be \( \Sigma \) the set of all possible content items \( \Sigma = \{i_1, i_2, ..., i_n, ...\} \), in each window time (day) we will deal with a subset \( \Omega \) of \( \Sigma \), \( \Omega = \{i_1, i_2, ..., i_y\} \) which is the set of items requested by the users. Furthermore, we define the vector \( R, R = \{r_1, r_2, ..., r_y\} \) which contains the number of times, each item has
been requested during the current window and the past cumulative number of requests of the same content. Each $r_i(k)$ can be expressed as follows:

$$r_i(k) = \sum_{j=1}^{k} r_i(j), \quad (4.1)$$

where $j$ is relative to the day before the current day when the item has been requested. Naturally, if the item has not been requested for a given day $x$, $r_i(x) = 0$, $k$ is a time variable. We define $W$ the set of Windows, $W = \{w(1), w(2), \ldots\}$. The training phase is performed for instance in $w(1)$ and the items in $w(2), w(3), \ldots$ are of two types: the newly requested, and the pre-cached resulting from prediction. Hence, the pre-cached items belong to $\hat{I}$, because they have been obtained after prediction. In addition, we consider that $w(k) = k$.

The predictor is inspired from [13]. The predictor uses one independent variable $\bar{r}(k)$ the average rating of items in the window time $w(k)$ (a day according to the cross-validation). This variable can be considered either global (average of all the request frequency of the item in the current window) or relative (average of request frequency assumed by the content during its life time). In summary, $\bar{r}(k)$ can be computed as follows:

$$\bar{r}(k) = \frac{\sum_{i=1}^{y} (r_i(k))}{|\Omega|} (\text{global}), k = 1, 2, 3, \ldots \quad (4.2)$$

where $|\Omega|$ is the cardinality of $\Omega$.

$$\bar{r}_i(k) = \frac{\sum_{j=1}^{k} (r_i(j))}{k} (\text{relative})k = 1, 2, 3, \ldots \quad (4.3)$$

Finally, parameter $b_i$ (popularity indicator or decision variable) intuitively added for lowering the MSE. It is defined as follows:

$$\hat{r}_{i11}^{i}(k) = \bar{r}(k) + b_{i11}^{i}(k), i = 1, 2, \ldots, y \quad (4.4)$$

$$\hat{r}_{i12}^{i}(k) = \bar{r}_i(k) + b_{i12}^{i}(k), i = 1, 2, \ldots, y \quad (4.5)$$

The equation (4.4-4.5) is an intuitive manner to define a decision variable $b$ that contains as much as possible information relative to a content $i$. This value is used to selecting item. $\hat{r}_{i11}^{i}$ uses the global average $\bar{r}$ and $\hat{r}_{i12}^{i}$ uses the local average $\bar{r}_i$.

The values $b_{i11}^{i}$ and $b_{i12}^{i}$ are determined by minimizing the residual error formulated through the convex optimization problem below:
\[
\begin{align*}
\min_{B_1^*} & \sum_{i=1}^{n} (r_i - \tilde{r}_{i1})^2, \\
\min_{B_2^*} & \sum_{i=1}^{n} (r_i - \tilde{r}_{i2})^2,
\end{align*}
\]  
(4.6)  
\[
\begin{align*}
\min_{B_1^*} & \sum_{i=1}^{n} (r_i - \tilde{r}_{i1})^2, \\
\min_{B_2^*} & \sum_{i=1}^{n} (r_i - \tilde{r}_{i2})^2,
\end{align*}
\]  
(4.7)

where \(B^*\) is the vector of \(b\) which minimize the equation (4.6-4.7). The election of item is done following the following rule. If after computation, \(b\) is positive and within a range to be determined, it means that the content item has high probability to be requested in the next running window; otherwise it does not. Without loss of generality, the above designed predictor is applied on every window time, which have been organized following the k-fold cross-validation described in Section 3.1.6.

However, this formulation suffers from over fitting. Over fitted because it gives good result when applied on the training data set and will not give good prediction when applying on the test data set. The mitigation of this effect can be done through regularisation. We are going to use the Tikhonov regularisation.

Tikhonov regularisation is the most commonly used method for regularisation, which is known in statistic as ridge regression. It consists on adding to the optimisation problem an extra expression. The generic regularised version of (4.6-4.7) is defined as follows:

\[
\begin{align*}
\min_{B^*} & \sum_{i=1}^{n} (r_i - \tilde{r}_i)^2 + \gamma^2 \sum_{i=1}^{n} (b_i)^2,
\end{align*}
\]  
(4.8)

where \(\gamma\) is the Tikhonov parameter. The most commonly value assigned to \(\gamma\) is the unity [13, 17]. Because it leads to get smaller value. Moreover, the regularisation reduces the conditioning of the problem, and gives direct solution. Hence, the final expression of (4.7) is:

\[
\begin{align*}
\min_{B^*} & \sum_{i=1}^{n} (r_i - \tilde{r}_i)^2 + \sum_{i=1}^{n} (b_i)^2,
\end{align*}
\]  
(4.9)

The value of \(b^i(k)\) obtained after resolution of the convex optimisation problem are:

\[
b_{11}^i = \frac{r_i - \bar{r}}{2},
\]  
(4.10)  
\[
b_{12}^i = \frac{r_i - \bar{r}_i}{2},
\]  
(4.11)
The vector $B^*$ collects $b^i$. The election of items is done by defining a threshold for selection of contents. For reason of fairness, i.e., give the same chance to every item to be selected one possible value of the threshold has been set to be equal to the average of $B^*$:

$$\Delta = \frac{\sum_{i=1}^{y} b^i}{y} \quad (4.12)$$

The items that might be requested in next window time are those which satisfied the inequality below:

$$b^i \geq \Delta, i = 1, 2, ..., y \quad (4.13)$$

Hence, the vector $\hat{I}$ (defined in Section 5.3.1) contains the set of elected items.

N.B: The above described predictor will be tuned in order to increase its accuracy. For instance we may add some parameter which takes into account the dynamic aspect related to the trendy of the item. For instance, an item which belongs to the category news may have the a number of requests decreasing over time. So, the predictor should not lose these substantial information.

An extension of the above predictor is done by adding a coefficient, which takes into consideration the probabilistic weight of the item among the set of items under analysis in the current window $k$. Let us assume that to have a uniform probability to be requested among the items in every window time, the new expressions of the predictors are given by

$$\hat{r}^{i_{21}}(k) = \bar{r}(k) + \Phi_i(k)b^{i_{21}}(k), i = 1, 2, ..., y \quad (4.14)$$

$$\hat{r}^{i_{22}}(k) = \bar{r}_i(k) + \Phi_i(k)b^{i_{22}}(k), i = 1, 2, ..., y \quad (4.15)$$

where

$$\Phi_i(k) = \frac{r_i(k)}{\sum_{i=1}^{y} r_i(k)}, i = 1, 2, ..., y \quad (4.16)$$

The corresponding value of $b^i$ after resolution of the convex optimisation problem is given by:

$$b^{i_{21}} = \frac{r_i - \bar{r}}{1 + \Phi_i}, i = 1, 2, ..., y \quad (4.17)$$

$$b^{i_{22}} = \frac{r_i - \bar{r}_i}{1 + \Phi_i}, i = 1, 2, ..., y \quad (4.18)$$

The value of $b^i$ in this case, considers feature relative to an item as additional information for tuning the predictor. The same method of selection is used.
However, the above defined threshold might be not sufficient for a good performance of the predictor. A second threshold relying on the content item frequency might be used. The predictor analysis will provide a guideline for the choice of the threshold.

4.2.2 Metric Specification

The above predictors predict the request (presence) of an item in the next window time. Hence, a suitable metric is to count the number of hit (Item requested the next day) and the number of miss (Item selected but not requested). The error of hit and miss is analysed in Chapter 5. Moreover, we have used interchangeably the two above predictors relying on their performance in order to reduce the hit error which is well explained in Chapter 5.

4.2.3 Linear Model 2

Similar to the previous predictor, this predictor relies only on one independent variable, and it is the simplest used in data mining. It has been borrowed from straight-line regression analysis [12]. The theory behind its mathematical formulation assumes that the predicted value (response variable) has constant variance. In addition, it is used for tracking and an item request stream. It is expressed by the below expression:

\[ r_i(k) = \alpha_0 + \alpha_1 \eta_i(k), i = 1, 2, ..., y \]  

(4.19)

Where \( \alpha_0, \alpha_1 \) are regression coefficients which may be estimated using the method of least squares, which allows to find coefficient which best-fit straight line and minimize the error between predicted and true value. And \( \eta_i \) is the inverse of the rank of the item which is used for defining a relation between the rank of an item and its request frequency. Hence, the estimate of coefficients is given by

\[
\alpha_1 = \frac{\sum_{i=1}^{n}(\eta_i - \bar{\eta})(r_i - \bar{r})}{\sum_{i=1}^{n}(\eta_i - \bar{\eta})^2},
\]

(4.20)

\[
\alpha_0 = \bar{r}_i - \alpha_1 \bar{\eta}_i
\]

(4.21)

where \( \bar{r}_i \) is the average value of the predicted value which are known. We are working on the training data, and the above predictor will be tested on the next days (test data). An extended model is developed further.
4.2.4 Metric Specification

We use as metric for performance analysis the prediction error, i.e., the difference between the predicted and true value as announced in Chapter 3. This model is just an attempt to track an item request frequency, and will be applied on the most requested content.

4.3 Regression Analysis and Kalman Analogy

Another mathematical model of the request stream relies on the concept of data fitting, which we formulate using the theory of regression analysis. Regression analysis is a powerful mathematical tool, which allows to establish a relationship between dependent variables (output) and independent variables (input) of a process. In addition, it is used for prediction and forecasting and overlap with machine learning. This model is just an approach for defining an formal mathematical model which matches with good approximation the content item request stream. Furthermore, based on the result obtained from the regression we are going to define a linear model and combine it with Kalman filter theory.

4.3.1 Regression Analysis

A non extensive analysis of content item life time through graphical plot has shown that their curve of rating may be approximated through a model. For that purpose, we have proposed a model of requested items stream based on regression analysis which uses the Generalize Linear Model, because we wish to model implicitly ordinal data. Furthermore, the regression matches our requirements because the properties of weak exogeneity, constant variance (homoscedasticity), independence of errors, and lack of multicollinearity are fulfilled by the data to mine. The property of exogeneity means that the data to mine do not suffer from some measurement error. Independence means there is not correlation between errors on independent variables. Homoscedasticity implies that response variables has the same variance in their error. Finally, lack of multicollinearity means that there is neither linear relation nor correlation between independent variables.

Let us consider the set of items of a data log denoted by $I = 1, 2, 3, \ldots, n$ and $r_i$ the total number of requests of the item $i \in I$. The proposed model is described by the equation below:

$$r^i(k + 1) = \beta_1 r^i(k) + \beta_0 \theta^i(k) + \epsilon^i(k), i \in I_k$$  \hspace{1cm} (4.22)
where \( r_i(k+1) \) is the rating value of item \( i \) in the next window time \( k+1 \). \( \beta_1 \) and \( \beta_0 \) are regression coefficients which may be used for minimizing the prediction error. The independent variable \( \theta^i(k) \) takes into consideration the effect of the popularity of the content item on its future solicitation by clients. Given that the independent variable uses the rank of the item \( i \in I(k) \) for impacting its future. Furthermore, it is an implicit expression of zipf-mandelbrot function [16]. \( I(k) \) is the set of items that have been requested during the window time \( k \). Finally, the variable \( \epsilon_i \) matches the effect of time on the life time of a content item which may introduce some error. In addition, we assume there is no correlation between the independent variables \( r^i, \theta^i \) and the noise \( \epsilon_r \). This assumption leads to unbiased and consistent properties of the estimator [17]. Here, \( k \) represents the window time of observation, which may be a day, couple of days, or a month. We have set it to a day. From the above formalism we define consequently the estimator model:

\[
\hat{r}_i(k+1) = \hat{\beta}_1 r^i(k) + \hat{\beta}_0 \theta^i(k), i \in I(k).
\]  

(4.23)

The residual error is defined by:

\[
e_i = r^i(k+1) - \hat{r}_i(k+1),
\]

(4.24)

where \( r^i(k+1) \) is the true value at time \( k+1 \) and \( \hat{r}_i(k+1) \) is the corresponding predicted. Let’s formulate the above equation in matrices:

\[
R(k+1) = R^T(k)B + E,
\]

(4.25)

where

\[
R(k+1) = \begin{pmatrix} r^1(k+1) \\ r^2(k+1) \\ \vdots \\ r^j(k+1) \end{pmatrix},
\]

\[
R^T(k) = \begin{pmatrix} r^1(k) & \theta^1(k) \\ r^2(k) & \theta^2(k) \\ \vdots & \vdots \\ r^j(k) & \theta^j(k) \end{pmatrix},
\]

\[
B = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix},
\]

\[
E = \begin{pmatrix} \epsilon \\ \epsilon \\ \vdots \\ \epsilon \end{pmatrix}.
\]
In this formalism, the vector $B$ is assumed to be constant in each window time. This reduces significantly the regression model and allows to have the above (4.24), matrices equation. In addition, the value of the matrices are retrieved from data mining of the provided data log. The noise is assumed Gaussian for simplifying the modelling. Such a system (m equations 2 unknowns, $j>2$) usually has no solution, so the goal is to find the coefficients $B$ which fit the equations “best,” in the sense of solving the quadratic minimization problem:

$$
\hat{B} = \arg\min_{\beta} (S(\beta)), \quad (4.26)
$$

where the objective function is defined by

$$
S(\beta) = \sum_{i=1}^{m} |r^i(k+1) - \sum_{i=1}^{n} R_{ij}\beta_j|^2 = \| R(k+1) - R^T(k)B \|^2, j = 1, 2. \quad (4.27)
$$

The choice of this objective function relies on some properties of the Least Square Estimator. The associated normal equation:

$$
\sum_{i=1}^{n} \sum_{t=1}^{2} R_{ij}R_{jt}\hat{\beta}_t = \sum_{i=1}^{n} R_{ij}R_{i1}(k+1), j = 1, 2 \quad (4.28)
$$

We use the Ordinary Least Squares (OLS) to determine the estimate coefficients, because it is both conceptually simple and computationally straightforward. This leads to the expression below of $\beta$:

$$
\hat{B} = (R^T(k)R(k))^{-1}R^T(k)R(k+1), \quad (4.29)
$$

Furthermore, Generalized Least Squares which is an extension of the OLS may be used for efficiently estimating of $B$. However, the error $\epsilon$ has to be well defined and matched the context of data analysis. The GLS relies on the feature of variability of the error on each requested content item. The method is general because it takes into account the existence of correlation or not between noise and independent variables; and response variables do not have same variance in their error. We consider that the error is relaxed, and i.i.d on any requested content item. In that case, the estimated value of $B$:

$$
\hat{B} = (R^T(k)\Omega^{-1}R(k))^{-1}R^T(k)\Omega^{-1}R(k+1), \quad (4.30)
$$

Where $\Omega$ is the covariance of the errors, which must be known.
4.3.2 Kalman Filtering Analogy

The life time of a content item can be considered as a trajectory (curve) that we are willing to track. From that perspective the Kalman filter theory can be of major use. The kalman filter is one the most consistent filter which allows to minimize the variance of the estimation error, and it can be applied to a system if and only if its dynamic may be described through a system of linear equations. The regression analysis model has been used for checking the feasibility in term of existence of linear law which describes the item request stream.

In the following, we study the accuracy of this model. So, in analogy to a linear dynamic system our content item request stream can be described by the following equations:

State equation:

\[ r(k + 1) = \beta_1 r(k) + \beta_0 \theta(k) + \epsilon(k), \]
\[ x(k + 1) = Ax(k) + Bu(k) + w, \]

Output equation:

\[ r_{\text{out}}(k) = \beta_2 \theta(k), \]
\[ y(k) = Cu(k) + z, \]

\( \beta_0, \beta_1, \beta_2 \) have the same meaning as mentioned in Section 4.4.1. In this context, for purpose of simplicity \( r \) is called the state of the system; \( \theta \) is a known input to the system; \( r_{\text{out}} \) is the measured output; \( \epsilon \) is the noise which fills the same meaning as the above described one. NB we deal with scalar quantities no vectors. The equations 4.31, 4.32, 4.33, and 4.34 show how we are adapting dynamic system behaviour to content item request stream.

We have measurement from the data log; can we determine the best estimate of the state \( r \). We want an estimator which provides values of the states that very little as possible from the true state. In order to match the Kalman filter theory, we assume that the noise component has average value zero, and there is not error in measurements. Hence, the noise can be modelled by a Gaussian distribution with null average and variance \( \sigma^2 \). Now, we can formulate the Kalman filter equations as follows [18]:

\[ K(k) = AP(k)C^T(CP(k)C^T + S_z)^{-1}, \]
\[ \dot{x}(k + 1) = (A\dot{x}(k) + Bu(k)) + K(k)(y(k + 1) - C\dot{x}(k)), \]
\[ P(k + 1) = AP(k)A^T + S_w - AP(k)C^TS_z^{-1}CP(k)A^T, \]

Where the \( K \) matrix is called the Kalman gain and the \( P \) matrix the estimate error covariance. \( \dot{x} \) is the state estimate. The second term in the equation
(4.36) is called the correction term and it represents the amount by which to correct the propagated state estimate. $S_w, S_z$ are respectively the process noise covariances and the measurement noise covariance:

$$S_w = E(w(k)W(k)^T),$$
$$S_z = E(Z(k)Z(k)^T).$$

In the context of content item request stream dynamic, we deal with scalars no vectors. Hence, the above equations can be re-written as follows:

$$g(k) = \beta_1 p(k) \beta_2 (\beta_2 p(k) \beta_2 + s_z)^{-1}, \quad (4.38)$$
$$\hat{r}(k+1) = (\beta_1 \hat{r}(k) + \beta_0 \theta(k)) + g(k)(r_{out}(k+1) - \beta_2 \hat{r}(k)), \quad (4.39)$$
$$p(k+1) = \beta_1 p(k) \beta_1 + s_e - \beta_1 p(k) \beta_2 s_z^{-1} \beta_2 p(k) \beta_1, \quad (4.40)$$

This Kalman adaptation does not suffer from error in measurements. Hence, $s_z = 0$ and the final Kalman equations for content item request stream are the following:

$$g(k) = \beta_1 p(k) \beta_2 (\beta_2 p(k) \beta_2)^{-1} \implies g(k) = \beta_1 \beta_2^{-1}, \quad (4.41)$$
$$\hat{r}(k+1) = (\beta_1 \hat{r}(k) + \beta_0 \theta(k)) + g(k)(r_{out}(k+1) - \beta_2 \hat{r}(k)), \quad (4.39)$$
$$p(k+1) = \beta_1 p(k) \beta_1 + s_e, \quad (4.42)$$

We remark that the gain is constant and depends on the coefficients $\beta_1$ and $\beta_2$. The coefficients of Kalman are those obtained during the regression analysis. We will illustrate these results by simulation analysis.
Chapter 5

Simulation and Results

We may evaluate the performance of the above described predictors. This work relies on the data log which contains 3 months of traffic from December 1994 to February 1995. In addition, the plot of the performance of all models developed in the previous chapters (Chapter 4) will be done in order to ease the visualisation of the information retrieved from the data log.

5.1 Experiments Setting

As above mentioned, we had to compress data log file for ease the content item tracking analysis. This preliminary work has been done using the programming language Java. Then, a linked list has been used for saving data. The format used for data storage is:

Item name (URL) | history

The field history contains the whole history of item organized following the format below:

day of request | total number of requests | rank out of the total number of items of the window.

We have seen in Chapter 4 how the contents request stream traffic plot look. The mining of data log focuses on images and video. I have used Matlab for the graphical representation of results. The Popularity of content (High request frequency) is also a fundamental item feature which might influence the performance of predictors. Figure 5.1 below shows the ranking dynamic of content item for the same above mentioned period. High values (curve with high y-axis coordinates) mean the item is not popular, and low values the
opposite. We notice that only items with spread dynamic and high request rate have important rank. Moreover, the depicted graph can be deduced from the multimedia content life time dynamic. The popularity of content is an inherent information that can lead to a logical prediction of future of that item. Nevertheless, there is not always a direct implication between popularity and future request prediction of the targeted item. That is why, a meticulous mathematical formalism have been conducted in order to design predictors which match significantly (low error) content item dynamic.

![Figure 5.1: Three months Item Ranking Dynamic Track.](image-url)
5.2 Performance Analysis

5.2.1 Predictor Model 1

The simulation and performance analysis develop in this section rely on the predictors designed in chapter 4 Section 4.2. The predictor type 1 refers to the predictor without the independent variable $\Phi$. Let us re-write their respective formulation:

\[
\begin{align*}
\hat{r}_{11}^i &= \bar{r} + b_{11}^i, i = 1, 2, ..., y \quad \text{(type 1.1)} \\
\hat{r}_{21}^i &= \bar{r} + \Phi b_{21}^i, i = 1, 2, ..., y \quad \text{(type 1.2)} \\
\hat{r}_{12}^i &= \bar{r}_i + b_{12}^i, i = 1, 2, ..., y \quad \text{(type 2.1)} \\
\hat{r}_{22}^i &= \bar{r}_i + \Phi b_{22}^i, i = 1, 2, ..., y \quad \text{(type 2.2)}
\end{align*}
\]

And their respective decisional variables are:

\[
\begin{align*}
b_{11}^i &= \frac{r_i - \bar{r}}{2}, i = 1, 2, ..., n \\
b_{12}^i &= \frac{r_i - \bar{r}_i}{2}, i = 1, 2, ..., n \\
b_{21}^i &= \frac{r_i - \bar{r}_i}{1 + \theta_i}, i = 1, 2, ..., n \\
b_{22}^i &= \frac{r_i - \bar{r}_i}{1 + \theta_i}, i = 1, 2, ..., n
\end{align*}
\]

Performance of Predictors Type 1.1 and 1.2

We have performed for each predictor a learning led by a cross-validation. We have considered the case with global average request frequency per window (equations in $\bar{r}$) for the computation of $b_i$. The election of each item has been done by setting the threshold $\Delta$ as mean value of the vector $B^*$. We have got the result below Figure 5.2.
Figure 5.2: Predictor type 1.1 performance (hit prediction plot and error on the left).

Figure 5.3: Predictor type 1.2 performance (hit prediction plot and error on the left).

The results are not satisfactory for neither predictor type 1.1 nor type
1.2. This is due to that the mean value of the vector of decisional variable might be negative. Hence, that gives chances even to the most irrelevant content items to be selected. Figure 5.4 below shows the trade-off in setting of the threshold.

![Figure 5.4: Threshold setting.](image)

We can lower the error by setting the threshold to half of the max of $b^j \in B^*$. The results are promising, Figure 5.5 5.6 depicts the experienced improvement.
Both predictors (type 1.1 and type 1.2) experience a low error variance.
The average error is 1. This result is satisfactory compared to the one where
the threshold has been set to the mean value of the vector $B^*$. In addition,
we notice that the number of hit predicted increases by 1 for the predictor
type 1.2 respect to the predictor type 1.1. The predictor type 1.1 has an error
more steady than the predictor type 1.2. However, by lowering the threshold
no lowering of the error is experienced consequently. The hit error becomes
higher. Figure 5.7 shows the the performance of the predictor 1.1, with $\Delta$
set to one quarter of the max $b^i \in B^*$.

![Performance of Predictors Type 2.1 and 2.2](image)

Figure 5.7: Predictor type 1.1 performance with $\Delta$ set to one quarter of the
max $b^i \in B^*$ (hit prediction plot and error on the left).

**Performance of Predictors Type 2.1 and 2.2**

Now, we consider the above equation in $\bar{r}_i$, i.e., the average request frequency
of content in each window time is computed relatively to the content. And
we are going to set the threshold as we have done during the Performance
analysis 1. Figure 5.10 illustrates the case with threshold $\Delta$ as mean value
of the vector $B^*$.
Figure 5.8: Predictor type 2.1 performance with $\Delta$ as mean value of the vector $B^*$ (hit prediction plot and error on the left).

Figure 5.9: Predictor type 2.2 performance $\Delta$ as mean value of the vector $B^*$ (hit prediction plot and error on the left).

We can infer from the plots above that $\Delta$ as mean value of the vector...
$B^*$ is not a good threshold for any type of predictors (global average or content relative average). However, there are more hit for these predictors (type 2.1 and 2.2) than the one with global average. Figure 5.11 depicts the performances for threshold $\Delta$ set to the half of $b^i \in B^*$.

Figure 5.10: Predictor type 2.1 performance with $\Delta$ set to the half of the max of $b^i \in B^*$. (hit prediction plot and error on the left).
Both predictors (type 2.1 and type 2.2) outperform the global average based one (type 1.1 and 1.2). They have same performances; it means that the parameter $\Phi$ does not influence the performances of the predictors. Moreover, the minimum hit ratio they achieve is 2. And the average hit error is lower than the average hit error of the predictor global average based (type 1.1 and type 1.2).

Figure 5.12 shows that lowering the threshold does not improve performances as well.
Figure 5.12: Any predictor (type 2.1 and and type 2.2) with $\Delta$ set to the quarter of the max of $b^i \in B^*$. (hit prediction plot and error on the left).

In summary, both predictors can not provide with perfect exactitude request frequency of content, but can provide an indicator of probable election of content. Moreover, we can infer from the plot of the prediction error and the plot of traffic that error varies with high frequency in period of intense traffic. That is due to that the variation of a content request stream can be sudden, i.e., it can have high request frequency one day and the next day an extremely low requests frequency; and it is more steady during low traffic period. This behaviour influences significantly the performances of predictors as shown on the plots. That influence becomes more evident for values of the threshold lower than the best value; which effect is shown on the above figure, in which the high fluctuations are observed either in period of low or intense traffic. The relative request frequency based predictors are better than the global one. They achieve a very low hit error ratio. However, they are not sensitive to the parameter $\Phi$. This out performing of the relative average content request frequency based predictor is due to the fact that the local average is lighter than the global, that gives more chances to the content to be selected.

The Tikhonov regularization is one the design component which has improved the results. However, the number of item elected is too small. It might be due to that our analysis have been conducted on non-recent data (December 1994 to February 1995). The performance analysis relying on
regression and Kalman filter is reported in next section.

5.2.2 Predictor Model 2

The mathematical formalization of this predictor have been done in chapter 4 section 4.3. Moreover, the performance analysis have been conducted on one of the popular contents. We have used the first 7 days of the month to computing the coefficients $\alpha_0$ and $\alpha_1$. And applied them for prediction for the remaining days of the data log.

Figure 5.13: Item request frequency track by predictor modelled in Section 4.3.

Figure 5.13 above depict the performance of the data mining provided linear model. The error varies with high frequency. In addition the predictor has good enough performance when the content has a low rank. We can infer that it has considerable probability to predict well for that situation. The performance is not good, but depending on the requirement of the CDN designer this model might be useful. This plot emphasizes on that design a mathematical model which tracks content request over time is not an easy task. The regression and Kalman filter are extension theory of this model.

5.2.3 Regression and Kalman Filter

We have attempted to use the Kalman filter in the context. The parameters of Kalman filter have been retrieved from regression analysis. We have used
for regression Ordinary Least Squares method and the General Least Squares. A linear regression has been used because it is a good candidate which matches the requirements of Kalman filter theory. The figures below illustrate the performances of each regression method and the corresponding error. Figure 5.14 depicts item track with OLS and the error. The effect of the time on item lifetime has been modeled with normal distribution, and we have bounded the values that the error can assume. In addition, the performances of Kalman’s filter have been conducted by tracing one of the popular contents.

The above Figure 5.14 depicts the performance of regression based on OLS. The result shows an impressive matching between the predicted request frequency and real value; which can be inferred from the dynamic computation of the regression coefficient. The calculation of the predicted value does not use regression coefficient already found, but performs a calculation online, that is to say, everything is done as the memory on the past is only based on the current value that is used to predict the future value (a leap into the future). This result shows how good the linear model matches our requirement and the tracking of an content item life time. Online also means that the computation of the regression coefficient is done using only natural value, i.e., request frequency and its rank without any additional component. The soul of the reasoning of this extended linear model seems to follow the Markov chain theory.

Figure 5.14: Item request frequency track by OLS.
Figure 5.15 illustrates the performances of the GLS. We notice that they are as good as the performances of the OLS. In addition, we can infer that linear model is good candidate for tracking a content request frequency over time. However, this result remain good until we just do one hop prediction, which is appropriate. We have seen how much the curve of life time dynamic of a content item has a complex behaviour. The regression coefficient obtained at each iteration of the OLS are plotted on Figure 5.16. We can infer from the graphs that the value of regression coefficients fluctuate with high frequency. Those variations are due the attempt to match the ongoing item request frequency by the regressor. Nevertheless, those values have been used for tracking item request frequency stream for Kalman filter, which performances are shown further.
Kalman’s filter adaptation have been done in Chapter 4. The related performance are shown through some plots. Firstly, we have considered the Coefficient of linear equation used for Kalman theory constant over time.
They have been got from the regression analysis. Figure 5.17 has been done with $\beta_2 = \beta_0$, $\beta_0$ set to the mean value of the vector $BOLS_0$, and $\beta_1$ set to the mean values mean ($BOLS_1$); where $BOLS_0$ and $BOLS_1$ are respectively the vectors of value of $\beta_0$ and $\beta_1$ obtained from OLS. And $\hat{r}(0) = r(0)$ as initial condition. The performance of Kalman’s filter are good but not enough for matching suitably the item life time track. For that purpose the previous model (hit and miss ratio) seems to give better results. The above theory has been done just for checking if Kalman’s filter theory might be a good candidate for prediction of content request frequency in CDN.
Chapter 6

Conclusion and Future Work

6.1 Goal and Insight

The main goal of this master’s thesis was to apply learning algorithms in Content Delivery Networks for improving the caching of content through the design of dynamic predictors. With such a goal in mind, some algorithms have been proposed in Chapter 4. Some tuning parameter which take into account as much as possible information inherent to a content have been used to making predictor 2 more consistent than predictor 1. The traffic analysis have been done using data from data log dated from January 1994 to February 1995.

We have designed two linear models for matching the content request frequency. We have seen in Chapter 4 that content request frequency follows a complex evolution over time, which makes tricky the prediction of exact request frequency. This suggest to use as metric for performance analysis for both modelled dynamic predictors the hit ratio. Moreover, for exploiting as much as possible information relative to contents, we have a modified method of cross-validation in order to strengthen the training process and improve the performance of the learner during the test phase. The definition of the threshold for decision making for content election has been an important challenge. And simulations have shown that the best value of the threshold is half of the max value of the vector of decision variables for both predictors. In addition, we have kept all the process dynamic by applying not only the learning online but the exploration methods, i.e., the threshold have been computed every single day of test after training in order to match the natural evolution of the traffic. An additional personal trick have been done which consist of cumulate the request frequency of item over time in order to keep track of all content and give them a fair chance to be selected. We have
also seen that the predictor type 1.1 is more stable than the predictor type 1.2 referring to predictors with global average. Because the predictor type 1.2 is more sensitive to the inherent information of a content than predictor 1.1. Moreover, the predictors with local average of content (predictor type 2.1 and 2.2) out perform the global average based one (predictor type 1.1 and predictor 1.2). However, in my opinion their (predictors) mathematical modelling might be improve by adding more suitable and well-defined parameters, which might lead to improve their performances.

Other models have been designed for tracking the request frequency of content. The challenge in defining a model was the definition of independent variables which match the context, and give significant insight in apprehending the substance of the researched goal. The performance of that predictor (model2) has been analysed in Chapter 5 Section 5.2.2. Base on its performance analysis, it might be tuned for improving its performances. Moreover, we have conducted an analysis for proving the validity of the linear predictor by designing an extended model, which results have been used for tracking content request frequency with Kalman's filter theory. The results have shown the suitability of the linear model and discard any analysis based on non-linear model. Kalman's filter performance analysis showed poor performance because, the noise introduced in the the mathematical modelling of the linear model matches more the features of Poisson distribution (Exponential memoryless property) than Gaussian. Moreover, Kalman has failed because request frequency stream of contents seems to be memoryless.

6.2 Future Work

The capability to predict content seems a promising area that might improve significantly the caching not only for CDN, but all other technologies involving knowledge and prediction of user behaviour through their browsing activities. In addition, the designed predictors can be tested in CDN simulated environment for further performance analysis.

6.3 Privacy Issue

Dynamic predictors for content selection in Content Delivery Network is a field which involves mining data of internet users, by which privacy remains an important issue to take into account. Hence, engineers working in that realm should not scrutinize or track users. Moreover, data have to be mined by strict confidentiality protocols, i.e., leak is not acceptable. The data log
mined in this master thesis has been obtained by respecting these protocols. The results have been presented without any specification of the origin of the data.
Bibliography


